Amendments to the Specification:

Please replace paragraph [026] with the following paragraph, marked to show changes:

FIG. 2 illustrates a typical optimizing compiler 20, comprising a front end compiler 24, a code optimizer 26, and a back end code generator 28. The front end 24 of a compiler takes as input a program written in a source language 22 and performs various lexical, syntactical and semantic analysis on this language outputting an intermediate set of code 32 representing the target program. This intermediate code 32 is used as input to the code optimizer 26 module which attempts to improve the intermediate code so that faster-running binary ~~machine~~ code 30 will result. The method and apparatus of the present invention offers improved optimization by providing improved updating of profile frequency for procedure inlining.

Please replace paragraph [036] with the following paragraph, marked to show changes:

The IPA-based inlining separates the inlining decision from the inlining transformation. Because the inlining decision is based on the call graph, it has global information about the program and thus can make better choices. However, the inlining decision must be saved so that the inlining transformation can finish transformation later based on the saved information. To represent which call is inlined into which procedure and through which callers, the Inlining Plan (IP) will be used. This technique is described in: Matthew Arnold, Stephen Fink, Vivek Sarkar, and P. F. Sweeney, "A Comparative Study of Static and Profile-based heuristics for Inlining," 52-64, ACM SIGPLAN DYNAMO'00, January 2000, which has been incorporated herein by reference for all purposes.